Practical Machine Learning - Prediction

Vincent Phan

December 24, 2015

Executive Summary

In this report, we will be applying machine learning algorithms to predict our desired outcomes. The data set we will be analyzing is collected by Groupware@LES. This data set is generated using an accelerometers placed on the participant's belt, forearm, arm, and the dumbbell while performing a unilateral dumbbell bicep curl in five different fashions. More detail of this experiment can be found here.

The goal of this report is to predict the manner in which the participant performed the exercise(classe). **Note that this report is fully reproducible**

```
library(dplyr); library(caret); library(rattle)
```

Loading packages

```
training_data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing_data<- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")

#changing all blanks to NAs
training_data[training_data ==""] <- NA
```

Reading orignal training & testing data set provided

Quick Exploratory Data Analysis This data set contains 159 variables we can use to predict classe with. Not all variables will be useful to make predictions as it will increase processing time. The **Cleaning Data** section show how to eliminate those unwanted variables.

```
dim(training_data)
```

```
## [1] 19622   160
```

Data Splitting During test phrase, I had created a model based on splitting **training_data** into 70% training and 30% training. Doing this had increased the accuracy by 1.15% (99.71% vs 98.56% below); However, doing this had increased processing time exponentially. I will be sacrificing accuracy for speed. Also, the 1.15% decrease in accuracy proved to be insignificant when predicting the **testing_data** set.

Below, we will divide the data into 33%. With this 33%(total 6479 observation vs before 19622), we will create a training set with 70% of the data.

```r
set.seed(777)
inTrain <- createDataPartition(training_data$classe, p=0.33, list=FALSE)
training <- training_data[inTrain, ]
dim(training)
```

```
## [1] 6479  160
```

```r
set.seed(777)
inTrain1 <- createDataPartition(training$classe, p=.7, list=FALSE)
train1 <- training[inTrain1,]
test1 <- training[-inTrain1,]
dim(train1); dim(test1)
```

```
## [1] 4537  160
```

```
## [1] 1942  160
```

Cleaning Data

- Using function nearZeroVar(), we can remove any predictor variables with variances close to 0.
- We will be removing any variables that contains 75% or more of NA.
- The 1st 5 variables seems to be irrelevant (time stamp, names & etc), we will be removing those as well.

```r
nearzero <- nearZeroVar(train1)
train1 <- train1[,-nearzero]

NAs <- sapply(train1, function(x) mean(is.na(x))) > 0.75
train1<- train1[,NAs==FALSE]

train1 <- train1[,-(1:5)]
dim(train1)
```

```
## [1] 4537   54
```

Training our data with different machine learning models

- We will be training our data with 3 different machine learning models: Random Forest(rf), Recursive Partitioning and Regression Trees(Rpart), and Learning Vector Quantization(LVQ).
- We will be pre-processing the data within the train function. Our data will be "center" and "scale" to optimize performance.
- We will be using simple cross validation with 4 folds.

```r
#RF Model
train1fit <- train(classe ~. ,data=train1, preProcess=c("center", "scale"),method="rf", trControl=train
predict_test1 <- predict(train1fit, test1)

#LVQ Model
```

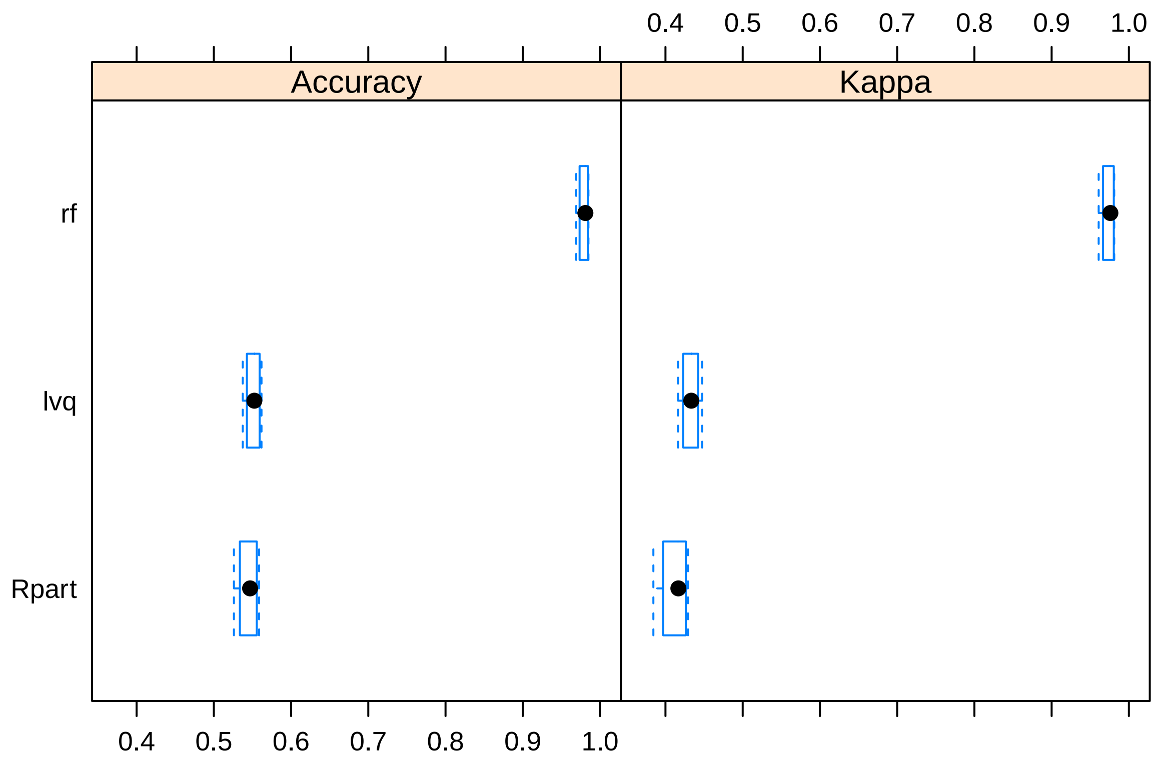
```
fit_lvq <- train(classe~. , data=train1, method ="lvq", trControl=trainControl(method = "cv", number = 
predict_test1_lvq <- predict(fit_lvq, test1)


#Rpart Model

fit_rpart <- train(classe~. , data=train1, method = "rpart", trControl=trainControl(method = "cv", numbe
predict_test1_rpart <- predict(fit_rpart, test1)
```

Comparing and selecting best model Random Forest Model proved to be highly accurate compared to the other 2 machine learning models: Max 98.50% compares to 56.17% and 56.78%. We will be using this method to predict our outcome in the test data set.

```
results <- resamples(list(rf=train1fit, lvq=fit_lvq, Rpart=fit_rpart))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: rf, lvq, Rpart
## Number of resamples: 4
##
## Accuracy
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## rf    0.9691  0.9758 0.9811 0.9791  0.9844 0.9850    0
## lvq   0.5374  0.5454 0.5525 0.5510  0.5581 0.5617    0
## Rpart 0.5260  0.5376 0.5470 0.5446  0.5540 0.5586    0
##
## Kappa
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## rf    0.9609  0.9693 0.9760 0.9735  0.9802 0.9810    0
## lvq   0.4163  0.4263 0.4335 0.4327  0.4398 0.4475    0
## Rpart 0.3844  0.4034 0.4167 0.4117  0.4250 0.4292    0
```

```
bwplot(results)
```



Look at our best model

- 500 trees
- 27 variables at each split
- 1.45% OOB estimated of error rate
- Looking at the plot, 200-300 trees is sufficent when building this model. Click here for plot http://i65.tinypic.com/29x85qb.png. The error rate seems to be flatting out between 200-300 trees. If we need to retrain our data, reducing the number of trees to 200-300 will help decrease processing time.(I could not get rmd to embed an image so I had included a link for the plot if you wish to review VIA tinypic.)

```
train1fit
```

```
## Random Forest
##
## 4537 samples
##   53 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (53), scaled (53)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 3401, 3404, 3404, 3402
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9720064  0.9645713  0.005074286  0.006427945
##   27    0.9790582  0.9734976  0.007334870  0.009293602
##   53    0.9717813  0.9642972  0.006511312  0.008240047
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Predicting our test1 data set with our best model With our best model, our accuracy rate is 98.56%.

```
pred <- predict(train1fit, test1)
confusionMatrix(pred, test1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 547   3   0   0   0
##          B   4 370   8   0   2
##          C   0   2 331   4   3
##          D   1   1   0 314   0
##          E   0   0   0   0 352
##
## Overall Statistics
##
##                Accuracy : 0.9856
##                  95% CI : (0.9792, 0.9904)
##     No Information Rate : 0.2842
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9818
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9909   0.9840   0.9764   0.9874   0.9860
## Specificity            0.9978   0.9911   0.9944   0.9988   1.0000
## Pos Pred Value         0.9945   0.9635   0.9735   0.9937   1.0000
## Neg Pred Value         0.9964   0.9961   0.9950   0.9975   0.9969
## Prevalence             0.2842   0.1936   0.1746   0.1637   0.1838
## Detection Rate         0.2817   0.1905   0.1704   0.1617   0.1813
## Detection Prevalence   0.2832   0.1977   0.1751   0.1627   0.1813
## Balanced Accuracy      0.9944   0.9876   0.9854   0.9931   0.9930
```

Out of sample error Our out-of-sample error is 1.44% (1-.9856). This model is highly accurate and we will be using this to predict our final final test set: **testing_data**.

```
predict_testing_data<- as.character(predict(train1fit, testing_data))
predict_testing_data
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

Result

The model that was built using random forest machine learning model proved to be highly accurate. I had submitted the homework assignment with 100% result. Thank you for reading.